Vimos vários aspectos da herança de protótipos durante a aula, vamos resumir alguns deles aqui e colocar alguns pontos extras:

Protótipo vs `prototype`

Praticamente todos os dados criados em JavaScript (objetos, arrays, etc) têm um protótipo, porém apenas alguns deles têm a propriedade `prototype`. São estes objetos que contêm a propriedade `prototype` que acessamos com `Object.prototype`, `Array.prototype` e etc durante o vídeo, que definem os protótipos para todos os outros acima na cadeia.
Ou seja, todos os objetos criados a partir de `Object`, `Array`, `String`, etc, têm um protótipo que foi "herdado" destes, mas não necessariamente têm uma propriedade `prototype`.

Cópia vs referência

Os métodos e propriedades **não** são copiados de um objeto para outro na cadeia de protótipos - eles são acessados pelo interpretador ao percorrer a cadeia e os métodos executados de acordo com o `this`, ou seja, o contexto em que o método foi executado.

`__proto__` será descontinuado

O uso de `__proto__` durante a aula é só para dar exemplo e acessar a cadeia de protótipos. Esta é uma propriedade "assessor" que serve para "expor" o protótipo dos objetos e está em processo de ser descontinuada. Você pode ler mais sobre esta propriedade na documentação do MDN.

Não altere o `prototype`

Não é recomendável alterar diretamente o `prototype`, pois alterar diretamente as regras de herança de qualquer objeto afeta a performance do código em qualquer interpretador, seja o NodeJS ou navegadores. Vamos ver como criar objetos de acordo com as boas práticas do JavaScript em seguida.

Cuidado com a performance

Todas as propriedades de uma cadeia de protótipos são enumeradas e o tempo que o interpretador leva para pesquisar uma propriedade, desde o nível mais alto na cadeia, pode ser longo e impactar o desempenho. Além disso, se o código tentar acessar uma propriedade

não existente, vai percorrer toda a cadeia durante a busca. Assim, não é uma boa prática criar longas cadeias de protótipos.